Niveau(x) d'enseignement: 4 Numéro EDUQ : 9570548
Centre d'intérêt: 640

Cloutier, Jean-François.

Structures informatiques du prototype tutoriel conceptuel (SOCRATIX-0.2) / J.-F. Cloutier, B. F. Rousse, H. D. Stolovitch. -- Montréal : Université de Montréal, Programme de perfectionnement des maitres en enseignement professionnel, 1985.
ii, 37 p. ; 28 cm.

Ss-titre: Rapport sur l'informatisation d'un tutoriel conceptuel universel intelligent : les structures informatiques du prototype SOCRATIX-0.2 & les fondements théoriques liés à la création d'un tel logiciel. "Rapport technique IV (82/MPS/02-PPMEP 10)". Résumé: p. 1.

Descripteur(s): *Enseignement assisté par ordinateur *Didacticiel *Informatique *Technologie éducative *Secondaire (Niveau)

Identificateur(s) de sujet: *Apprentissage de concepts

Le document décrit les structures informatiques minimales requises pour l'élaboration d'un didacticiel d'enseignement de concepts, tel qu'utilisé pour le prototype "SOCRATIX-0.2" et fournit les bases théoriques nécessaires au développement d'un système didactique conceptuel informatisé (SDCI) exploitant l'ensemble des possibilités du modèle cognitiviste de l'enseignement.

"SOCRATIX-0.2" a été réalisé et soumis à un processus d'expérimentation en laboratoire et dans plusieurs écoles. Les structures informatiques du prototype sont décrites. Ses limites inhérentes permettent de définir des bases théoriques pour la description des structures informatiques minimales d'un tuteur conceptuel universel intelligent (TCUI).

"SOCRATIX-0.2" est un programme de type classique, c'est-à-dire algorithmique. Le programme est composé de plusieurs modules résidant en mémoire, soit en permanence, soit seulement lorsqu'ils sont requis. "SOCRATIX-0.2" utilise la banque de concepts comme source de données et produit un fichier contenant les résultats diagnostiques de l'interaction avec un apprenant.

Type de document: r
Financement: Bénéficiaire - Université de Montréal
Organisme(s) source(s) - MEQ-DGERU
Programme(s) de subvention - PPMEP

EDUQ 8570548

STRUCTURES INFORMATIQUES DU PROTOTYPE TUTORIEL CONCEPTUEL (*SOCRATIX-0.2*)

(RAPPORT SUR L'INFORMATISATION D'UN TUTORIEL CONCEPTUEL UNIVERSEL INTELLIGENT, LES STRUCTURES INFORMATIQUES DU PROTOTYPE SOCRATIX-0.2 & LES FONDEMENTS THEORIQUES LIES A LA CREATION D'UN TEL LOGICIEL)

CLOUTIER J-F.
ROUSSE B.F.
STOLOVITCH H.D.

MARS 1985

Rapport Technique IV

(82/MPS/02-PPMEP 10)

PROGRAMME DE PERFECTIONNEMENT DES MAITRES EN ENSEIGNEMENT PROFESSIONNEL

UNIVERSITE DE MONTREAL * FACULTE DES SCIENCES DE L'EDUCATION

SOCRATIX.TIT: 21/03/85

EDUQ 8570548

STRUCTURES INFORMATIQUES DU PROTOTYPE TUTORIEL CONCEPTUEL (*SOCRATIX-0.2*)

(RAPPORT SUR L'INFORMATISATION D'UN TUTORIEL CONCEPTUEL UNIVERSEL INTELLIGENT, LES STRUCTURES INFORMATIQUES DU PROTOTYPE SOCRATIX-0.2 & LES FONDEMENTS THEORIQUES LIES A LA CREATION D'UN TEL LOGICIEL)

CLOUTIER J-F.
ROUSSE B.F.
STOLOVITCH H.D.

MARS 1985

Rapport Technique IV

(82/MPS/02-PPMEP 10)

PROGRAMME DE PERFECTIONNEMENT DES MAITRES
EN ENSEIGNEMENT PROFESSIONNEL

UNIVERSITE DE MONTREAL * FACULTE DES SCIENCES DE L'EDUCATION

SOCRATIX.TIT: 21/03/85

NOTE

Ce texte fait partie des documents produits dans le cadre du projet de recherche PPMEP(82/MPS/02), réalisé entre janvier 1983 et mai 1985, ayant pour titre:

"ELABORATION & EXPERIMENTATION D'UN MODELE DIDACTIQUE INFORMATISE POUR L'APPRENTISSAGE DE CONCEPTS TECHNOLOGIQUES ABSTRAITS CHEZ DES ETUDIANTS DU SECTEUR PROFESSIONNEL".

Cette recherche a pu être menée grâce à une subvention du Ministère de l'Education du Québec (PPMEP).

Ce projet a été supervisé et dirigé par:

. Harold STOLOVITCH, *Directeur du projet et Professeur agrégé en Technologie éducationnelle, Université de Montréal.*

. Viateur LEMIRE, *co-irecteur du projet et Responsable du programme PPMEP, Faculté des Sciences de l'éducation, Université de Montréal.*

A:NOTEDIT

Table des matières

Résumé 01

CHAPITRE I- L'informatisation de SOCRATIX. 02

1. SOCRATIX-0.2 04

2. Un prototype limité. 05

3. La banque de données 08

- *Le problème de la représentation graphique* . 08

- *Organisation des données décrivant un concept* 11

- *EDITCON: l'éditeur de la banque de concepts.* 14

CHAPITRE II- Description du prototype SOCRATIX-0.2 . . 18

1. Principes de base gouvernant l'interaction . . 20

- *Economie des types de réponses demandées de l'usager.* 21

- *Contrôle du rythme de présentation* 23

- *Accès facultatif à l'information supplémentaire.* 24

- *Séparer information opérationnelle et information didactique* 25

2. Organisation modulaire de Socratix 0.2 27

- *La procédure-maîtresse* 27

- *Le module DEPART* 29

- *Le module CIRCUIT.* 30

- *Le module ENSEIG* 33

- *Le module TESTE.* 34

- *Le module CENSEIG.* 35
- *Le module CTESTE* 36
- *Le module FINAL.* 37

* * * * *

RESUME

Ce document décrit et commente l'informatisation de Socratix, un tuteur conceptuel universel intelligent. Prototype expérimental limité, Socratix-0.2 a été réalisé et soumis à un processus d'expérimentation en laboratoire et dans plusieurs écoles.

Les structures informatiques du prototype sont décrites et ses limites inhérentes permettent de définir des bases théoriques pour la description des structures informatiques minimales d'un tuteur conceptuel universel intelligent (T.C.U.I.).

L'objectif d'univertsalité de l'expertise tutorielle de Socratix a déjà amené à la nécessité de l'élaboration d'une théorie de la représentation informatique des concepts en réseau (cf. A.R.C. - en cours de développement)(1).

* * * * *

(1)- logiciel d'aide à l'Analyse de Réseaux Conceptuels

CHAPITRE I
L'INFORMATISATION
DE SOCRATIX

L'objectif du projet "*Socratix*" est la programmation d'un tuteur ayant les caractéristiques suivantes:

1- Une expertise en didactique des concepts;

2- Une expertise diagnostique (capacité de générer un modèle cognitif adéquat de l'apprenant);

3- Une expertise stratégique (capacité d'adapter le niveau d'enseignement et les objectifs d'apprentissage de façon dynamique);

4- Applicable à tout domaine de contenus, pourvu que ces contenus soient des concepts.

Il fallait donc automatiser ces différentes expertises (*les programmer*) et, pour ce faire, elles devaient être d'abord explicitées, puis traduites dans un formalisme de représentation se prêtant à un traitement automatique.

Le substrat de ces expertises est le concept. L'expertise didactique se réfère inévitablement aux propriétés des concepts dans sa formulation. Il en va de même pour l'expertise diagnostique. L'expertise stratégique, quant à elle, se réfère surtout aux conclusions des précédentes.

Puisque le tuteur se veut universel (*indépendant de tout contenu*), les propriétés d'un concept ne doivent pas dépendre de la dénotation (*ou signification*) de ce concept. En fait, ces propriétés seront structurelles; elles se rapporteront au type d'organisation de l'information constituant le concept.

Voilà donc un aperçu sommaire du problème informatique que posait la réalisation de Socratix.

A l'intérieur du projet, un modèle didactique a été mis au point et a été testé grâce à deux prototypes limités du tuteur conceptuel (*Socratix-0.1 & 0.2*). Une théorie de la représentation des concepts en réseau a ensuit été développée.

1. SOCRATIX-0.2

Plusieurs recherches ont démontré l'efficacité de l'utilisation de "*séries rationnelles*" dans l'enseignement des concepts. Pourtant, ces recherches ont principalement, jusqu'à maintenant, expérimenté cette méthode didactique dans le cadre restreint de l'apprentissage d'un seul concept "*isolé*" ou de quelques concepts "*associés*". Cependant, un concept n'est pas une unité de connaissance "*fermée*"; sa compréhension dépend de la compréhension préalable des concepts qui participent à sa définition. Il restait donc à amplifier la méthode didactique de façon à tenir compte des pré-requis d'apprentissage (*la hiérarchie didactique*) et à tester cette méthode a de poursuivre plus avant. (*Pour une discussion plus complète, se référer aux documents de B. F. Rousse*).

De plus, la problématique d'un tuteur conceptuel informatisé est à ce point vaste qu'il est nécessaire d'appliquer cette stratégie commune au développement des systèmes informatiques complexes: d'abords construire un prototype limité qui exhibe un bon nombre des propriétés souhaitées du produit final, puis le tester intensivement afin d'en définir les limitations prévues et imprévues. Il s'agit là d'une phase exploratoire nécessaire quand l'étendue du problème à

résoudre est imparfaitement connue.

Finalement, les résultats accumulés lors de l'expérimentation du prototype qu'est Socratix 0.2 (*de mai 1984 à mai 1985*) permettent de substancier et d'émettre plusieurs hypothèses importantes concernant la méthode didactique employée (*c.f. B.F.Rousse*). L'expérimentation de mai 1984 s'est faite avec la première version du prototype (*Socratix 0.1*). Dès octobre 1984, la nouvelle version substantiellement modifiée et améliorée a été utilisée.

2. Un prototype limité

Pour l'élaboration de Socratix 0.2 plusieurs paramètres ont été fixés. L'objectif d'apprentissage final, ainsi que l'ordre de présentation des concepts pré-requis sont prédéterminés. Le niveau de langage et le degré de spécialisation demandés de l'apprenant sont eux-aussi prédéterminés. Le prototype ne construit pas de modèle cognitif de l'apprenant se contentant de faire un diagnostic immédiat de ses erreurs (*sous-généralisation, sur-généralisation ou confusion*) et d'offrir un traitement remédiatif conséquent. Enfin, le mode de présentation est limité au texte écrit et à la représentation graphique. Néanmoins l'élaboration de ce prototype s'est avéré une tâche d'une dimension appréciable.

Le domaine de contenu choisi pour l'expérimentation de Socratix (*versions 0.1 & 0.2*) est celui des TYPES DE CIRCUITS ELECTRIQUES. Il s'agit du concept de CIRCUIT ELECTRIQUE et de ses SOUS-CONCEPTS CIRCUIT ELECTRIQUE ELEMENTAIRE, EN SERIE, PARALLELE et MIXTE (*qui sont donc des concepts associés*). A ces concepts s'ajoutent les concepts prérequis de CONDUCTEUR ELECTRIQUE, SOURCE ELECTRIQUE, RESISTANCE ELECTRIQUE et de TRAJET ELECTRIQUE, ainsi que leurs représentations graphiques. L'objectif final était la connaissance des types de circuits électriques.

Socratix-0.2 présente les concepts autres que les types de circuits électriques (*les pré-requis*) de façon abrégée. Puis les quatre types de circuits sont présentés selon l'algorithme du modèle didactique. Ce modèle, en fait, est double, puisque deux procédures sont disponibles. La première, dite "*normale*", présente chaque concept indépendamment et séquentiellement. La seconde, dite en "*cascade*", présente les quatre concepts associés conjointement (*deux par deux*) en faisant ressortir leurs différences de fççon comparative. Le choix de la méthode est pré-fixé pour chaque apprenant, pour des raisons expérimentales (*pour une discussion détaillée, voir l'étude de B.F.Rousse*).

Le choix des types de circuits électriques comme domaine de contenu pour l'expérimentation a deux justifica-

tions. La première découle d'un sondage auprès de professeurs et d'étudiants du secteur secondaire professionnel, lequel place l'électricité parmi les sujets les plus problématiques, tant à enseigner qu'à apprendre (...) (*cf. le rapport de L.Rapagna*). La deuxième est d'ordre représentationnel: les types de circuits électriques se prêtent bien à une représentation graphique simple et efficace.

La première étape de la réalisation de Socratix-0.2 a été la création d'une banque de concepts pour les types de circuits électriques. Cette banque contient les définitions de chaque concept, ainsi que l'information graphique et verbale nécessaire à Socratix-0.2 dans sa démarche didactique. La mise au point de cette banque a nécessité la création d'un langage graphique et d'un logiciel (*EDITCON*) permettant de créer et d'éditer la banque de concepts.

La deuxième étape fut la programmation comme telle de Socratix-0.2. Socratix-0.1 a été, suite à une première expérimentation, au mois de mai 1984, considérablement amélioré. Toutes les informations qui suivent se rapportent à la nouvelle version 0.2 (*octobre 1984*). Néanmoins, un aperçu des modifications majeures apportées à la première version (*0.1*) se trouve à l'appendice A.

Les activités comportent la mise au point de la banque de connaissances d'un petit système-expert. Ceci permettra une étude approfondie des problèmes d'analyse informatique des contenus.

3. La banque de données

L'environnement informatique, un micro-prolog augmenté, permet déjà l'exploration interactive de l'expertise. Il faudra néanmoins augmenter la qualité de son interface pour permettre l'interprétation de requêtes en langage courant et la communication des réponses aussi en langage naturel.

3.1- Le problème de la représentation graphique

Pour des raisons de compatibilité avec le matériel disponible dans les écoles et de performance minimale, il a été décidé d'implanter Socratix-0.2 sur le PC d'IBM, avec 128K de mémoire, un lecteur de disques double face et écran graphique noir-et-blanc, comme configuration minimale.

Puisque les attributs des types de circuits électriqus sont présentés de façon graphique (*forme du circuit, continuité des conducteurs, nombre de trajets, nombre de résistances par trajet, etc..*), il était essentiel que le graphisme soit clair et précis. Par exemple, les diagonales

devaient être de vraies diagonales, pas une série de traits brisés (*ce qui entre autre aurait interféré de façon grave avec la perception de l'attribut de continuité dans un circuit*).

Aucun logiciel de génération graphique disponible sur l'IBM PC n'offrait conjointement une qualité graphique acceptable et la possibilité d'intégration à un programme écrit dans un langage de haut niveau.

Il a donc fallu avoir recours à l'emploi de caractères graphiques et créer un langage graphique intégrable à l'éditeur de concepts et au tuteur.

Un caractère graphique est semblable en tout à un caractère alphanumérique (*les lettres, chiffres etc.*) sauf que, contrairement à ces derniers qui sont prédéfinis, il est possible de les définir par programme. Un caractère est défini par huit rangées de huit points codés numériquement (*illustration 1*).

Faire du graphisme avec des caractères graphiques est l'équivalent d'écrire un texte avec comme lettres des parties de dessin (*illustration 2*).

Après une analyse des constituants des dessins de circuits électriques, un ensemble de caractères graphiques a été mis au point. A l'aide de ces caractères il est possible de construire les différent dessins de circuits nécessaires à Socratix 0.2.

Un peu comme on construit un mot à partir de lettres, des composantes de circuits électriques ont été définies par l'assemblage de ces caractères: ce sont les macros graphiques (*illustration 3*). Un langage graphique a été développé, non seulement pour représenter ces macros, mais aussi pour utiliser macros et caractères afin de dessiner des circuits électriques. Le langage permet aussi de faire de l'animation et est capable de mélanger texte et dessin (*voir appendice B pour une description formelle du langage graphique*).

L'interprète pour ce langage a été écrit en Pascal. Il a été intégré et à l'éditeur de banque de concepts et à Socratix 0.2.

Soixante-huit caractères graphiques et cent-vingt-deux macros ont été créés. Le langage graphique permet de les utiliser en mode graphique haute-résolution (*80 caractères par ligne et 24 lignes*) avec comme résultat des graphiques très lisibles et très propres (*voir appendice C pour la liste des caractères graphiques*).

Le langage graphique est d'application générale, par contre les caractères et les macros graphiques ont été spécifiquement conçus pour représenter une classe restreinte de circuits électriques. Ce langage ainsi que l'éditeur de concepts ont été créés dans le seul but de permettre l'élaboration de Socratix 0.2; ils n'ont ni la prétension d'être facile d'utilisation pour le non-initié ni celle d'être des outils d'application générale. Ils se sont par contre avérés essentiels comme outils de développement pour Socratix 0.2.

3.2- Organisation des données décrivant un concept

A chaque concept sont associées des données qui lui sont spécifiques. Une partie de ces données DEFINISSENT le concept: ce sont les attributs critiques et les attributs variables avec leurs champs de variation. Des données supplémentaire concernent l'information nécessaire aux mécanismes d'interaction avec l'apprenant: certaines concernent l'ensemble du concept, d'autres sont spécifiques à chaque attribut critique.

Voyons l'organisation d'un concept dans la banque de concepts que maintient le programme EDITCON:

1- INFORMATIONS GLOBALES:

1.1 Le NOM du concept;

1.2 Le NOMBRE de ses ATTRIBUTS CRITIQUES;

1.3 Le NOMBRE de ses ATTRIBUTS VARIABLES;

1.4 Le NOMBRE de paires d'exemple et non-exemple;

1.5 Le NOMBRE de paires dévolues à la série didactique;

(i.e. utilisées pour l'acquisition du concept; le même nombre est réservé pour tester l'acquisition et le reste sert de paires supplémentaires disponibles lors de l'acquisition comme supplément facultatif);

1.6 Message qui MOTIVE l'apprentissage du concept;

(la partie spécifique au concept du message standard utilisé par le tuteur pour motiver l'apprentissage du concept);

1.7 Message qui RAPPELE les concepts déjà appris et qui sont pré-requis à l'apprentissage du concept.

2- INFORMATION PRESENTE POUR CHAQUE ATTRIBUT CRITIQUE:

2.1 Le NOM de l'attribut critique;

2.2 Le message signalant la PRESENCE de l'attribut critique dans un exemple du concept;

2.3 Le message signalant l'ABSENCE de l'attribut critique dans un non-exemple du concept.

3- INFORMATION PRESENTE POUR CHAQUE ATTRIBUT VARIABLE:

3.1 Le NOM de l'attribut variable;

3.2 Le domaine de variation de l'attribut variable;

(Il est à noter que ceux-ci ont été soumis à plusieurs révisions plus ou moins importantes tout au long de l'élaboration de Socratix 0.2.)

4- POUR CHAQUE EXEMPLE DE CHAQUE PAIRE (*d'exemple et non-exemple proche*):

4.1 La valeur de chaque ATTRIBUT VARIABLE;

(valeur documentaire seulement)

4.2 La description en langage graphique de l'exemple;

4.3 La description graphique de la MISE EN EVIDENCE pour chaque attribut critique.

(séquence d'animation)

5- POUR CHAQUE NON-EXEMPLE DE CHAQUE PAIRE (*d'exemple et non-exemple proche*):

5.1 L'index de l'attribut critique MANQUANT;

(le non-exemple ne diffère de l'exemple que par l'absence d'un seul attribut critique)

5.2 La valeur de chaque ATTRIBUT VARIABLE;

(valeur documentaire seulement)

5.3 La description en langage graphique du non-exemple;

5.4 La description graphique de la MISE EN EVIDENCE pour CHAQUE attribut critique - présent ou manquant;

(séquence d'animation)

3.3- EDITCON: l'éditeur de la banque de concepts

Le banque de concepts, de sa création à son état final, a subi de nombreuses modifications et améliorations. Dans la première version de Socratix (*0.1*), les concepts étaient au nombre de 12: les 4 types de circuits électriques et les concepts pré-requis en électricité.

Maintenant, les concepts pré-requis n'ont plus le même statut que les types de circuits électriques. Ils ne sont pas soumis au traitement didactique complet tel qu'élaboré dans le modèle didactique; ils reçoivent un traitement abrégé et leur présence dans la banque de concepts n'est plus requise, du moins dans l'état actuel du prototype.

La première banque comportait une quantité substancielle d'information qui subissait fréquemment des transformations importantes. La nouvelle banque, plus modeste (*2*), a été elle aussi l'objet d'un grand nombre de manipulations. Un éditeur de banque de concepts a donc été un outil essentiel. L'investissement considérable (*en efforts*) qu'a constituée la programmation de cet éditeur s'est avéré finalement très rentable.

L'éditeur s'est montré surtout rentable dans l'expérimentation et la mise au point des séquences d'animation qui accompagnent CHAQUE attribut critique de CHAQUE membre de CHAQUE paire d'exemple et non-exemple. L'éditeur permet de tester très facilement plusieurs façons de coder dans le langage graphique, soit le dessin d'un exemple d'un concept, soit la mise en évidence d'un attribut critique de ce concept.

L'éditeur est commandé par des menus donnant accès à différentes options. Celles-ci permettent de:

1. *Lister les noms des concepts mis en banque*

2. *Vérifier ou modifier la hiérarchie didactique*

   (l'ordre dans lequel le tuteur présentera les concepts)

3. *Ajouter un nouveau concept à la banque*

   (cela se fait interactivement)

4. *Enlever un concept de la banque et de la hiérarchie didactique*

5. *Modifier n'importe quelle partie d'un concept*

   (de façon interactive et avec l'aide de sous-menus)

Plusieurs options sont offertes pour modifier un concept. On peut notamment modifier:

1. *son information GLOBALE (voir plus haut);*

2. *l'information sur les ATTRIBUTS CRITIQUES;*

3. *l'information sur les ATTRIBUTS VARIABLES;*

4. *l'information sur les SERIES de paires:*

   (par paire individuelle ou par série complète: c'est ici que se retrouvent les descriptions en langage graphique)

L'éditeur comprend de plus un compilateur et un décompilateur pour le langage graphique. Lorsqu'une description graphique est donnée à l'éditeur, celui-ci la traduit dans une forme différente, qui est alors exécutée par l'interprète graphique. Cette forme intermédiaire permet une exécution très rapide des descriptions graphiques. La première version de Socratix exécutait la forme non-traduite, avec un niveau de performance assez médiocre.

Avec un nouvel interprète et un compilateur, la deuxième version possède des performances graphiques excellentes tout en conservant la flexibilité du langage graphique.

Lors de l'édition des descriptions graphiques, ce processus de compilation est totalement invisible, puisqu'un décompilateur permet d'offrir à l'usager les descriptions graphiques, dans leur forme originale, c'est-à-dire celle sous laquelle elles ont été entrées.

Avant d'aborder la construction du prototype tutoriel, il a donc fallu décider de la représentation informatique d'un concept, créer des outils graphiques efficaces et construire un éditeur de banque de concepts flexible. Ceux-ci ont été soumis à plusieurs révisions plus ou moins importantes, tout au long de l'élaboration de Socratix-0.2.

* * * * *

CHAPITRE II
DESCRIPTION DU PROTOTYPE SOCRATIX-0.2

Socratix-0.2 est un programme de type classique, c'est-à-dire algorithmique. Par contre, il se distingue des didacticiels-types ayant cours encore actuellement par plusieurs caractéristiques importantes. Sa distinction primordiale est lié à l'utilisation d'un modèle didactique conceptuel supporté par une théorie valide de l'apprentissage et qui a été l'objet de nombreuses recherches et expérimentations. Ce modèle, de plus, est universel dans le sens où il s'applique à tout contenu conceptuel.

Le prototype possède une capacité diagnostique beaucoup plus précise et effective que celle trouvée sur les didacticiels-types; ceci est une conséquence directe de l'emploi du modèle didactique. Il sera démontré que le prototype n'exploite pas encore tout le potentiel diagnostique du modèle et que des structures informatiques plus évoluées que celles employées pour sa construction actuelle sont nécessaires à la pleine exploitation de ce potentiel.

De plus, des considérations de base, dans le design du type d'interaction entre le tuteur et l'utilisateur, distinguent encore Socratix-0.2 du didacticiel moyen.

Socratix-0.2 est écrit en Pascal MT+ (*Digital Research*) et compilé en langage machine pour exécution sous système opérationnel CP/M-86, sur un IBM/PC ou tout micro-ordinateur compatible-IBM. Les ressources minimales utiles ont été ramenées à 128K de mémoire centrale (*RAM*), un seul lecteur de disques et un écran graphique couleur ou monochrome.

Le programme est composé de plusieurs modules résidant en mémoire, soit en permanence, soit seulement lorsqu'ils sont requis. Ceci fut motivé par les dimensions du programme et le peu de mémoire centrale disponible sur les appareils utilisés pour l'expérimentation.

Socratix-0.2 utilise la banque de concepts comme source de données et produit un fichier contenant les résultats diagnostiques de l'interaction avec un apprenant.

Le prototype est capable d'appliquer deux types de procédures didactiques: une procédure dite NORMALE et une autre dite EN CASCADE. Lors d'une session complète, une seule de ces procédures est appliquée.

Une première analyse des structures informatiques de Socratix-0.2 abordera les principes généraux qui ont guidé le design des mécanismes d'interaction entre l'usager et le tuteur. Dans un deuxième temps, l'organisation modulaire du programme sera suivie, afin de dégager la structure logique du programme même. Par contre, une analyse subséquente abordera le problème des structures informatiques sur un plan conceptuel plus large, qui permettra de comprendre les limites intrinsèques du prototype et alimentera une discussion théorique des structures informatiques minimales d'un tuteur conceptuel universel intelligent.

1- Principes de base gouvernant l'interaction

Une considération de base dans la conception du prototype a été de limiter au maximum la diversité des entrées que doit effectuer l'usager au clavier. Ensuite nous avons choisi de laisser le rythme de présentation de la matière sous le contrôle de l'usager. De plus toute information supplémentaire à l'information didactique minimale n'est donnée qu'avec l'accord de l'usager. Enfin une distinction est faite dans la présentation entre l'information opérationnelle et l'information didactique.

1.1 Economie des types de réponses demandées de l'usager

Le tuteur requiert de l'usager une connaissance minimale du clavier. Lorsque le tuteur demande une réponse de l'usager, il lui est offert un choix très simple d'options. Voici les choix de réponses auxquels qui lui seront présentés:

Oui ou Non : *Presser sur la touche "O" ou la touche "N"*

A ou B : *Presser sur "A" ou "B"*

Signal pour continuer : *Presser sur la barre ESPACE*

Entrer un chiffre : *Presser sur "0", "1" ...*

Bien sûr, le tuteur ne fait pas de distinction entre une majuscule et une minuscule.

Si on fait exception de la procédure d'initialisation où l'usager doit donner son nom et presser sur la touche "*retour*", ces touches sont les seules qu'il doit apprendre à trouver.

S'il se trompe, le tuteur ignore totalement la touche pressée. Une erreur d'entrée ne peut endommager, ni le programme, ni l'apparence de l'écran. Une erreur n'élicite non plus aucun message de réprimande sonore ou écrit. La conséquence d'une mauvaise entrée, pour l'usager, est simplement que le tuteur l'ignore complètement et attend une entrée valide. Ceci évite un stress tout à fait inutile chez l'usager.

Il faut souligner que lorsqu'une réponse esr demandée de l'usager, le tuteur donne explicitement les noms des touches valides.

Ces principes somme toute très simples et très faciles à informatiser ont été extrêmement efficaces. Tous les élèves qui ont participé à la première expérimentation se sont familiarisés de façon quasi-immédiate avec le mode de communication entre eux et le tuteur. Ceci a été fortement facilité par la standardisation maximale des messages demandant une réponse. Après quelque temps, les élèves semblaient ne plus avoir à lire, mais reconnaître de façon immédiate ce qu'on attendait d'eux.

Il est d'ailleurs souhaitable que ces élèves (*du professionnel court notamment*) n'aient à faire des efforts de lecture que lorsque de l'information didactique pertinente leur est présentée (*du fait de leurs difficultés relatives de lecture*).

1.2 Contrôle du rythme de présentation

Lorsque de l'information statique est donnée à l'écran, le tuteur attend toujours un signal de l'usager avant de continuer. Par contre, lorsqu'il s'agit d'information dynamique, c'est-à-dire lors de la mise en évidence par animation graphique d'un attribut critique, cette information est redonnée répétitivement, jusqu'à ce que l'usager donne le signal pour la stopper et passer à la suite. Pour donner ce signal, l'usager n'a seulement qu'à appuyer sur n'importe quelle touche du clavier (*le tuteur suggère d'utiliser la barre ESPACE*).

Il y a quelques exceptions à ces règles, mais elles ne s'appliquent que dans des cas où l'information concernée a déjà été présentée plutôt au cours de la session.

1.3 Accès facultatif à l'information supplémentaire

A plusieurs moments, le tuteur offre à l'usager de l'information dont il POURRAIT avoir besoin. L'usager décide de recevoir ou non cette information. Par contre toute information jugée critique par le tuteur est donnée obligatoirement.

Par exemple, le modèle didactique est appliqué de façon complète, mais minimale. L'usager devra traverser ses différentes étapes. Dans chaque étape l'information minimale est communiquée. Dans plusieurs étapes l'usager peut recevoir de l'information redondante si il JUGE que cela lui est nécessaire. Parfois cette information lui sera nécessaire, néanmoins il sera seul responsable de sa décision d'y avoir accès ou non.

Choisir de laisser le maximum de responsabilités entre les mains des élèves est certes courir un certain risque. Par contre lors de l'expérimentation en laboratoire (*mai '84*) nous avons été fortement impressionés par le sérieux et l'opiniâtreté de TOUS les élèves participants, ainsi que par la valorisation personnelle qu'ils ont retiré de leur collaboration. Plusieurs ont ouvertement apprécié le type de contrôle qu'ils avaient pu exercé sur la pertinence de l'information redondante que pouvait leur fournir le tuteur.

D'autres facteurs sans doute plus fondamentaux ont contribué au comportement extrèmement positif des participants, néanmoins cet aspect du tuteur n'est sûrement pas un élément explicatif négligeable.

Si nous nous sommes permis d'élaborer sur un principe très simple, c'est qu'il nous apparaît intéressant de constater qu'un principe si élémentaire et se soumettant à une informatisation triviale puisse avoir des répercussions aussi importantes sur le succès d'un didacticiel.

1.4 Séparer information opérationnelle et information didactique.

Deux types d'information sont présentées à l'usager: des instructions guidant l'interaction entre lui et le tuteur et des connaissances qu'il doit apprendre. Ces deux types d'information diffèrent d'une façon fondamentale; l'information didactique demande un effort de mémorisation tandis que l'information opérationnelle ne demande qu'une action.

Il importe donc que l'usager les distingue bien et de façon immédiate. Une stratégie possible est de donner l'information opérationnelle verbalement (*par une interface sonore*) et l'information didactique par écrit. Dans le cas de Socratix 0.2, il était impossible d'utiliser ces deux médiums; il n'était pas possible non plus d'utiliser deux écrans ce qui aurait été une solution avantageuse à bien des points de vue.

L'option restante était de diviser l'écran en deux zones bien distinctes et de confiner autant que possible chaque type d'information sa partie d'écran respective.

Une petite fenêtre au bas de l'écran affiche les instructions que doit suivre l'usager et le reste de l'écran présente le matériel didactique. Le haut de l'écran est invisiblement divisé en deux zones, gauche et droite. Lorsqu'un schéma d'un exemple d'un concept est dessiné à l'écran et qu'il est accompagné du texte donnant les attributs critiques, le dessin apparaît dans une moitié et le texte dans l'autre. Lorsqu'une paire de schémas est présentée, chacun occupe une moitié.

En conservant le plus possible la même organisation de l'écran, on diminue l'effort que doit fournir l'usager afin de repérer et d'interpréter l'information.

Tous les principes qui décrivent les caractéristiques des interactions entre le tuteur et l'usager sont en fait des principes simples qui font appel à des considérations d'économie.

2- Organisation modulaire de Socratix 0.2

La structure logique de l'ensemble du programme est réflètée dans l'organisation modulaire des parties. Par contre si aucun graphisme n'est employé, il n'y a pas de restriction sur le domaine de contenu qui peut être représenté dans la banque de concept et, subséquemment, enseigné par les modules ci-haut mentionnés.

2.1 la procédure-maîtresse.

La procédure-maîtresse voit à l'instanciation de quelques variables globales (*nous y reviendrons*) et appelle le module d'initilisation DEPART. Ensuite, le tuteur amorce l'interaction avec l'apprenant, à l'étape suivant immédiatement la dernière étape qu'il a préalablement complètement terminée, et selon la méthode didactique (*NORMALE ou EN CASCADE*) qui lui a été pré-assignée.

En prenant pour acquis que l'étudiant en est à sa première interaction, nous allons suivre l'ordre d'exécution des modules. Le module DEPART se charge de l'initialisation générale du tuteur et est responsable du protocole d'identification de l'étudiant, qui a lieu à chaque activation du tuteur. Puis le module CIRCUIT enseigne les concepts prérequis de façon abrégée.

Ce qui suit dépend du type de procédure didactique attribuée à l'étudiant.

S'il s'agit de la procédure NORMALE, pour chacun des concepts dans la banque de concepts, deux modules sont activés en séquence. Le premier est le module ENSEIG, qui voit à l'acquisition du concept, et est suivi du module TESTE, qui vérifie son acquisition.

Par contre, s'il s'agit de la procédure EN CASCADE, tous les concepts en banque qui sont des concepts associés sont présentés conjointement par le module CENSEIG et leur acquisition conjointe est testée par CTESTE.

L'étape finale, dans un cas comme dans l'autre, est la présentation d'un test cumulatif, avec le module FINAL.

2.2 Le module DEPART.

Le module DEPART exécute trois grandes procédures. La premiére définit les caractères graphiques. La deuxième prépare Socratix-0.2 à avoir accès à l'information sur disquette. Cette procédure informe, entre autre, Socratix-0.2 du nombre de concepts en banque, de la hiérarchie didactique qui décrit l'ordre dans lequel ils doivent être présentés, crèe un fichier pour mettre les résultats du diagnostique si l'étudiant n'en a pas encore, trouve le type de procédure didactique (*NORMALE ou EN CASCADE*) qui sera utilisée et vérifie si l'étudiant a déjà eu accès au tuteur sur sa disquette, ainsi que l'endroit où il est rendu dans sa leçon.

La troisième procédure est responsable du protocole d'entrée que doit suivre l'étudiant, afin de savoir son nom et de vérifier, grâce à un code, s'il s'agit d'un étudiant qui débute ou du même étudiant ayant déjà travaillé avec la présente disquette. Dans ce dernier cas, il est indispensable que l'utilisateur donne le bon code. Ce mécanisme très grossier de protection apparaît suffisant pour ce type de didacticiel, dans l'état actuel.

Cette procédure a aussi l'avantage de préparer l'utilisateur au type d'interaction qu'il rencontrera tout au long du didacticiel. L'écran qui le guide dans sa démarche d'initialisation est divisé en deux parties: une boîte, au bas de l'écran, contient l'information instruisant l'utilisateur de ce qu'on attend de lui et le choix d'options qui lui sont offertes. La signification de cette région au bas de l'écran lui est communiquée lors de l'initialisation.

2.3 Le module CIRCUIT.

Ce module contient une série de procédures dont l'exécution enseigne de façon abrègée les différents concepts pré-requis à l'apprentissage des concepts représentant les quatre types de circuits électriques. Ces procédures répondent au besoin d'abréger la leçon autant que possible; Socratix-0.1 enseignait TOUS les concepts, pré-requis et autres, avec la méthode didactique complète, ce qui était trop long pour les besoins expérimentaux.

Une procédure abrégée fut donc mise au point pour Socratix-0.2, spécifiquement pour les concepts pré-requis aux types de circuits électriques, afin de fournir à chacun l'information minimale préalable requise. Cette procédure a été codée DIRECTEMENT et le code du module est donc totalement spécifique au contenu qu'il décrit.

A l'inverse, la méthode didactique codée dans les modules ENSEIG, CENSEIG, TESTE et CTESTE ne dépend absolument pas d'un contenu précis. Ces modules s'appliquent, avec quelques restrictions mineures, à tout concept qui a pu être entré dans la banque de concepts. C'est la configuration de la banque de concepts qui restreint le domaine du contenu, puisque le langage graphique utilise des caractères et des macros graphiques spécifiques aux circuits électriques. Par contre, si aucun graphisme n'est employé, il n'y a aucune restriction sur le domaine de contenu pouvant être représenté dans la banque de concepts et, subséquemment, enseigné par les modules ci-haut mentionnés.

Lors de l'exécution du module, un premier message motive l'apprentissage des concepts pré-requis en informant l'utilisateur de leur nature de concepts pré-requis à l'apprentissage des types de circuits électriques.

Le prochain message donne la nature FONCTIONNELLE du circuit électrique: c'est un circuit où circule de l'électricité.

L'étape suivante consiste à enseigner les composantes principales des circuits électriques : les conducteurs, résistances et les sources électriques. Pour chaque composante, le module donne sa fonction à l'intérieur du circuit électrique, quelques exemples courants et sa représentation

graphique. La repésentation graphique d'une composante est communiquée en lui donnant un ensemble varié de symboles représentant cette composante. On lui laisse le soin de percevoir le champ de variation possible pour une représentation d'une composante. Par exemple, pour la source électrique, les symboles varieront dans leur orientation spatiale, dans le nombre traits entre les deux pôles de la source et dans la polarité de la source (*voir illustration*).

Une fois que les représentations graphiques des composantes électriques ont été enseignées, une procédure montre un exemple de représentation de circuit électrique. L'utilisateur est alors à même de voir comment conducteurs, résistance et source sont intégrés dans un schéma de circuit électrique.

Une animation simple symbolise ensuite le passage du courant dans le circuit faisant le lien entre la FONCTION du circuit électrique et sa représentation schématique.

Le prochain concept a être abordé est celui de TRAJET ELECTRIQUE. La définition conceptuelle de trajet est donnée et des exemples de circuits à un seul ou plusieurs trajets sont donnés. Une animation simple décrit le comportement du courant. La façon dont l'attribut critique NOMBRE DE TRAJETS sera mis en évidence lorsque requis est démontrée.

Finalement la définition du concept CIRCUIT ELECTRIQUE est donnée. Suivent un exemple et un non-exemple de circuit électrique.

Ceci complète la description du module CIRCUIT. Ce module ne pouvait que subir une analyse descriptive de son comportement puisque, tant dans sa conception que dans son informatisation, il ne répondait qu'à des objectifs de contenus spécifiques.

2.4 Le module ENSEIG.

Le module ENSEIG est responsable de la procédure didactique dite NORMALE. Deux étapes précèdent la phase à proprement parler didactique. La première étape consiste simplement en un message qui MOTIVE l'apprentissage du nouveau concept: le message donne la pertinence d'un tel apprentissage.

La deuxième étape fait le RAPPEL des pré-requis: un message cite les concepts préalablement appris qui sont nécessaire à l'apprentissage du nouveau concept.

La phase didactique comporte, dans l'ordre, la présentation des ATTRIBUTS CRITIQUES illustrés à l'aide d'un exemple-type, la présentation d'une SERIE RATIONNELLE au cours de laquelle on tente d'augmenter les habiletés de discrimination et de généralisation de l'usager, et une série complémentaire facultative. (*Pour le détail du modèle didactique, se référer au document de B.F. Rousse*).

Les procédures du module ENSEIG sont générales et sont applicables à tout concept qui peut être mis dans la banque de concepts. En effet, elles n'utilisent, en fait d'informations de contenu, que celles présentes dans la banque de concepts.

Pour chaque concept enseigné, le module ENSEIG enregistre le temps qu'a mis l'usager en mode d'acquisition et le nombre de paires supplémentaires qu'il a demandé.

2.5 Le module TESTE.

Les procédures du module TESTE vérifient l'acquisition du concept précédemment enseigné par le module ENSEIG. Pour ce faire, il donne une série rationnelle nouvelle en brisant préalablement les paires et en présentant les exemples et non-exemples dans un ordre aléatoire. L'usager doit distinguer les exemples des non-exemples.

Selon ses réponses pour chaque paire (*le tuteur se souvient à quelle paire appartient chaque item*), un diagnostique est établi. A la fin du test, l'usager voit son score et peut, s'il le désire, voir ses erreurs et les raisons pour lesquelles il aurait dû faire un autre choix de réponse.

Utiliser une série rationnelle pour tester l'acquisition d'un concept permet de porter un diagnostique de sous-généralisation, de sur-généralisation ou de confusion (*s'il y a lieu*).

Finalement le diagnostique est mis sur fichier.

2.6 Le module CENSEIG.

La procédure didactique EN CASCADE est éxécutée par le module CENSEIG. Les concepts désignés comme associés sont présentés ensemble dans une formule où ils sont mis en contraste réciproquement (*voir le document de B.F.Rousse*).

La procédure initiale prévue consistait à présenter les quatre types de circuits électriques ensemble sur le même écran. A cause de la résolution graphique moyenne du IBM-PC, un maximum de deux concepts pouvaient être présentés simultanément. Il a donc fallu faire un compromis en présentant les concepts par paires successives (*ELEMENTAIRE-*

SERIE, SERIE-PARALLELE, PARALLELE-MIXTE).

Pour chaque paire, on définissait le DEUXIEME concept, puis on mettait les deux concepts en opposition. Afin d'amorcer la procédure en cascade, il fallait donc définir préalablement le concept de CIRCUIT ELECTRIQUE ELEMENTAIRE en utilisant la procédure "*normale*". Le nombre de cascades est égal au nombre de paires de la série rationnelle didactique d'un concept associé (*ils en ont tous le même nombre*).

Le module enregistre le temps mis par l'usager à l'intérieur du mode acquisition et le nombre de cascades supplémentaires qu'il a demandé.

2.7 Le module CTESTE.

Ce module teste l'acquisition conjointe des concepts enseignés par le module CENSEIG. Des exemples nouveaux de chacun des concepts associés sont présentés un à un, dans un ordre choisi de façon aléatoire. L'usager doit identifier le concept dont l'exemple présenté est un élément.

Les réponses sont enregistrées et le score final est donné à l'usager, qui peut alors, s'il le désire, obtenir l'explication de ses erreurs.

Le tuteur ne peut pas, dans ce cas, faire de diagnostic précis à partir d'une seule série d'exemples de chacun des concepts associés (*voir B.F.Rousse*). Ainsi, seules les réponses sont mises sur fichier.

2.8 le module FINAL.

Un test final est automatiquement administré. Il est essentiellement identique au test donné par CTESTE, sauf qu'en plus d'exemples de chacun types de circuits électriques des non-exemples de circuits électriques sont aussi présentés. L'usager doit identifier correctement chacun des items nouveaux qui lui sont présentés.

Les réponses sont enregistrées sur fichier et l'usager reçoit son score. Il n'y a pas possibilité pour l'usager de revoir ses erreurs.

A la complétion du module, l'usager reçoit un message de félicitation et le tuteur termine son éxécution.

* * * * *